Final Technical Report

NAG 2-834

(5/1/93 to 6/30/95)

Inductive Approaches to Improving Diagnosis and Design for Diagnosability

Douglas H. Fisher (Principal Investigator)

Box 1679, Station B
Department of Computer Science
Vanderbilt University
Nashville, TN 37235

615-343-4111

Vanderbilt University
Division of Sponsored Research
512 Kirkland Hall
Nashville, TN 37240

1 Introduction

Research funded by NAG 2-834 has followed three primary directions, and a number of secondary directions as well. The first primary areas of research carried out by Manganaris and Fisher (1994) and Manganaris (1995a, 1995b, in press), address the problem of classifying time series according to their morphological features in the time domain. A supervised learning system called CALCHAS, which induces a classification procedure for signatures from preclassified examples, was developed under NAG 2-834 funding. For each of several signature classes, the system infers a model that captures the class's morphological features, using Bayesian model induction and the minimum message length approach to assign priors. After induction, a time series (signature) is classified in one of the classes when there is enough evidence to support that decision. Time series with sufficiently novel features, belonging to classes not present in the training set, are recognized as such. Section 2 describes CALCHAS is somewhat greater detail and reports results from experiments in a monitoring domain of interest to NASA – the EGIL data.

A second primary area of research by Ortega and Fisher (1995) assumes two sources of information about a system: a model or domain theory that encodes aspects of the system under study and data from actual system operations over time. A model, when it exists, represents strong prior expectations about how a system will perform. Our work with a diagnostic model of the RCS developed by Peter Robinson at NASA Ames motivated the development of SIG, a system which combines information from a model (or domain theory) and data. Robinson's model tracks actual RCS data over time, and determines what of several *a priori* known operating modes (e.g., normal, pressure regulator failed closed or failed open) are consistent with the system's actual behavior. As it tracks RCS behavior, the model computes quantitative (e.g., the derivative of pressure) and qualitative (e.g., consistency of an operating mode with data) values. Ortega's work treats these computed values as additional high-level features that are used to augment the original data. Induction is then performed over the data represented by both the 'raw' features and the model-computed high-level features.

The following sections elaborate on CALCHAS, SIG, and some basic clustering research, which were the core of our research efforts. After these descriptions we very briefly summarize secondary work on other learning

strategies and other application areas that indirectly stemmed from our work in the two primary areas.

2 CALCHAS: Induction over Temporal Data

Early NAG 2-834 funded research by Manganaris, Fisher, and Kulkarni (1993) was concerned with the discovery of fault and normal operating modes of the RCS from data. The framework adopted by this work segmented continuous data streams (e.g., RCS telemetry data), and encoded the segmented data in a manner appropriate for a machine induction system. Using a very simple strategy of segmenting data streams into uniform-sized intervals, finding linear models for data segments, and clustering data segments using an unsupervised learning system, Manganaris, Fisher, and Kulkarni discovered behavioral patterns corresponding to normal and faulty operating modes. They also used discovered pattern for diagnosis, and obtained good diagnostic accuracy.

This preliminary work suffered from several limitations. First, the basic clustering procedure has been significantly improved by Fisher (1995, 1996); this includes a method of *iterative optimization* that appears novel in the clustering literature. We return to this strand later. A primary direction that stemmed from our work with the RCS was concerned with developing a more sophisticated segmentation and segment-modeling strategy. Thus, Manganaris and Fisher (1994) developed CALCHAS, which implements a strategy for fitting temporal streams with piecewise polynomial models using minimum description length principles.

2.1 The Calchas Task

Performance improvement in classification tasks has been a traditional area of machine learning. The objects to be classified are usually described by time-invariant attribute values. This research effort was motivated by applications in temporal and sequential domains. In such domains, an object's properties often vary with time; objects are described by a *time series of values* for each attribute.

This effort focuses on learning to classify time series based on the morphological features of their behavior over time (i.e., the shape of their plots).

Here we focus on the simplest case, where induction is performed on univariate time series (i.e., each object is described by one time-varying attribute). The term signature will be used synonymously with the term univariate time series.

2.2 Induction of Class Models and Classification

A set of preclassified signatures (the training examples) are presented to CALCHAS simultaneously. Given that signatures in the same class share morphological characteristics, the system infers *class models*, represented by functions of time, that capture them. Functions in the space considered by CALCHAS can be decomposed into a set of polynomials and intervals, with one polynomial per interval. For example, Figure 1 shows a signature and the class model induced from it. A Bayesian model induction technique finds the function best supported by the training data (Cheeseman, 1990). For each class, the system searches for the model M with maximum posterior probability in light of prior information I and training data D.

$$P(M|D,I) = P(M|I)\,\frac{P(D|M,I)}{P(D|I)} \tag{1}$$

To assign priors, $P(M|I)$, the minimum message length approach (Rissanen, 1983; Wallace & Freeman, 1987) is used. The negative logarithm of the prior probability of a model, $-\log_2 P(M|I)$, is equal to the theoretical minimum length of a message that describes M in light of prior information I. A very similar and influential technique in the design of CALCHAS has been used for surface reconstruction in computer vision (Pednault, 1989); another related technique has been exploited for learning engineering models to support design (Rao & Lu, 1992).

Class models are parameterized, thus the search for the best model extends in the space of parameters. CALCHAS uses the parameters in (Pednault, 1989) and an additional precision parameter. Each class model has a partitioning of the time domain into a sequence of intervals. For a given interval a search is made through all possible families of parameterized models; we use polynomials of up to degree two, but, the method can be easily generalized. To facilitate probabilistic predictions, we assume a Gaussian noise model and independence of sampling errors. We also assume that the variance of the noise distribution is constant over an interval. For each interval CALCHAS

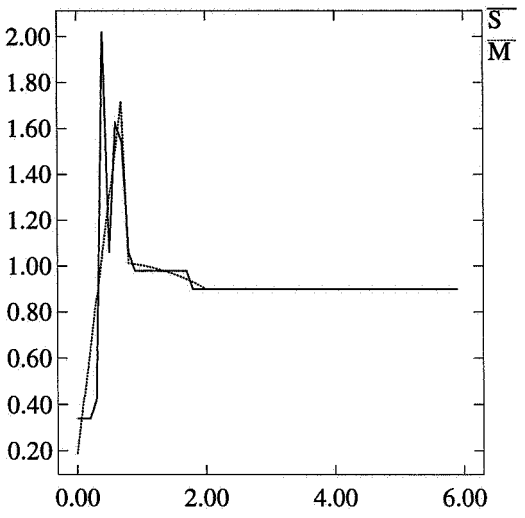

Figure 1: A signature (S) and the class model induced from it (M).

estimates the coefficients of the polynomial and the variance of the noise that maximizes the posterior probability of the model.

After training, given a signature, S, and a set of class models, the goal is to find the model most likely to be correct for the signature in light of the prior knowledge. We treat this as a hypothesis testing problem: for each class, C, we compute the *evidence*, $e(C|D,I)$, that S is an object of the class C (Jaynes, 1993):

$$e(C|D,I) = 10\log_{10}\left[\frac{P(C|D,I)}{P(\overline{C}|D,I)}\right] \tag{2}$$

The probability that S belongs in a class other than C, $P(\overline{C}|D,I)$, is computed from the posterior probabilities of all other classes and from the posterior probability of a special "novel" class. The likelihood of the "novel" class is set to zero when any of the known classes has a non-negligible likelihood. When all known classes have low likelihoods, its likelihood is computed so that it tends to one as the maximum likelihood among the known classes tends to zero. The prior of the "novel" class is set to an arbitrary low value. Under normal circumstances, the "novel" class plays no role in the compu-

tation of evidence, because of its very low posterior. Only when all known classes have low posterior probabilities, does the "novel" class become a viable alternative.

2.3 A Monitoring Application

The Electrical Generation and Integrated Loading (EGIL) controllers at NASA monitor telemetry data from the Shuttle to detect various events that take place onboard. Typically, an event is the onset or termination of operation of an electrical device on a power bus. Each event has a signature with a set of distinguished morphological characteristics, based on which the controllers identify them. There are over two hundred different events of interest, making their accurate identification a challenging task.

Signatures are extracted from the telemetry stream whenever a change in one of the currents is detected that exceeds a preset threshold. All signatures have the same duration (6 sec. after the triggering change), and their baselines are normalized by subtracting a suitable DC value.

We designed a set of experiments to demonstrate the feasibility of automating the classification of EGIL signatures using CALCHAS. Here we focus on the effect of training in classification performance. We use the percentage of correctly classified instances as our dependent measure of learning. In our experiments there are ten classes of signatures for ten different events; the average number of signatures per class is about 65. Our current implementation only handles univariate time series. There are many three-dimensional signatures in the EGIL domain; in these cases we ignore two of the phases.

In each run, we train CALCHAS on an equal number of randomly selected signatures from each class. We then evaluate its performance on the remaining signatures. We vary the amount of training by using different training set sizes. The results with training sizes of one and eight are summarized in the *confusion matrix* shown in Table 1. Each entry of the table shows the percentage of test signatures, in the class labeling the row, that were classified by CALCHAS to the class labeling the column. The top row for each class was obtained after training CALCHAS with one signature per class; the bottom row was obtained with training sizes of eight. All percentages are averaged over twenty runs; the standard deviations are shown. For example, with a training set of eight signatures, an average of 74% of the WCS test signatures were correctly classified as WCS, and 1% and 25% were incorrectly

Table 1: Classification of EGIL signatures (assumed univariate—see text).

CLASS		PHO	VAC	AWCS	H2O	CAB	PRP	WCS	TPS	RCR	GAL	NOVEL
PHO	1	40±29			1±4				2±7		57±29	
	8	96±5									4±5	
VAC	1		68±32									32±32
	8		93±2									7±2
AWCS	1			92±22	5±22							3±1
	8			96±2								4±2
H2O	1	2±9			98±9							
	8				100±0							
CAB	1					79±17						22±17
	8					90±16						10±16
PRP	1						98±4				2±4	
	8						98±2				2±2	
WCS	1							52±28		1±0		47±28
	8							74±4		1±0		25±4
TPS	1	7±14							76±17	3±5	15±11	
	8	8±7							85±8			7±7
RCR	1						2±0			97±1		
	8						3±0			97±0		
GAL	1	2±1									98±0	
	8	22±40									78±40	
UN1	1	46±10			13±2		12±2		3±2	2±1	22±9	2±0
	8	55±4			12±1		12±3		1±1	3±1	15±7	2±0
UN3	1	9±5			20±4		30±4		8±4	4±1	9±3	20±0
	8	18±2			15±1		29±2		11±2	4±1	3±2	20±0

classified as RCR and NOVEL, respectively. In general, the matrix diagonal indicates the percentage of correct classifications. Entries corresponding to UN1 and UN3 are for signatures whose actual class was unknown.

Table 1 indicates that increased training results in higher classification accuracies. A notable exception seems to be the GAL class, where training with eight signatures results in significantly lower accuracy than training with one signature. We suspect that GAL is an example of a disjunctive concept: there is more than one pattern of morphological features describing signatures in the class. When these experiments were performed, CALCHAS was unable to handle disjunctive concepts; training on multiple patterns for a class resulted in a confused class model and thus lower classification accuracy. However, this limitation has recently been eliminated with a version of CALCHAS that does learn disjunctive concepts (Manganaris, 1995b).

Beyond the practical advantages of automatic versus manual monitoring, a Bayesian learning approach offers the following technical advantages. It provides a principled way of discerning the distinguishing features of a

signature from measurement noise; it mitigates the problem of overfitting. CALCHAS provides an estimate of the confidence in each classification. When more than one classification is supported by roughly the same evidence, we can recognize this fact and report it, as opposed to making an arbitrary classification. Similarly, we can report when no classification is supported with significant evidence. Signatures with sufficiently novel features, belonging to classes not present in the training set, are recognized as such and are classified as NOVEL; potentially costly classification mistakes are avoided.

2.4 Future Work

As noted, Manganaris recently extended CALCHAS to learn 'disjunctive' class models. The initial system assumed that a class of like-events could be well described by a single pattern (piecewise polynomial pattern). However, in some cases time series belonging to the same general class have very different patterns (i.e., corresponding to disjunctive concepts). For example, in trying to distinguish normal walkers from those with cerebral palsy, Manganaris's initial system would try to characterize sample gaits in each class by a single temporal pattern. Unfortunately, the muscle 'activations' of normal walkers may exhibit very different temporal patterns. Disjunctive class models mitigate the limitation of the earlier system in an important way. In addition, the event signatures in the EGIL domain are often defined over multiple channels; Manganaris is extending CALCHAS to deal with the multichannel case.

3 Exploiting Models and Data

When human expertise is nonexistent or very weak relative to a particular domain/task, and when data is plentiful, machine induction from data may be the only reasonable approach to task automation; this was the philosophy taken with CALCHAS. In contrast, when expertise is strong, then encoding the expert's model or domain theory via traditional knowledge acquisition strategies may be the best approach. In fact, this human expertise may stem from induction over a much larger data sample than is available at the time task automation is undertaken.

In many cases, however, conditions are indeterminate as to whether sole

reliance on machine induction or human expertise is most appropriate: human expertise may not be 'perfect' and/or data may not be as plentiful as desired. In cases where some data is available and human expertise is less than perfect, an advantageous strategy may be to exploit both in an appropriate way.

Drastal, Czako, and Raatz (1989), Rendell and Seshu (1990), and Ortega (1994) suggest a strategy that loosely couples empirical learning and model-based reasoning: the data is augmented by 'features' that are actually intermediate terms of the domain theory and which are deemed true of a datum by deductive application of the domain theory. Induction is then performed over this augmented data set. If domain-theory-derived features are included in rules derived inductively, then this suggests a rough consistency between the model and data; model features may be viewed as somewhat better predictors than 'raw' features because noise is mitigated. If model features are not referenced in a resultant classifier, this may speak to imperfections in the model and/or this behavior may stem from an unrepresentative data sample. In both cases model-derived features may not look as informative as 'raw' features relative to the *available* data.

SIG is a system that augments data with domain-theory-derived 'features', but unlike previous work, this system biases an adaptation of C4.5 (Quinlan, 1993) to select domain-theory based features even when this conflicts somewhat with C4.5's original bias to select the most 'informative' feature as computed over the data. The intent is to guard against the possibility of unrepresentative data. However, the domain-theory preference bias may be overridden if C4.5's original bias is sufficiently opposed to the domain-theory preference bias. The intent here is to acknowledge that there may be some imperfections in the domain theory.

3.1 SIG: Motivation

SIG was motivated by our attempts to inductively build classifiers of faults of the Reaction Control System (RCS) of the Space Shuttle. A mixed qualitative/quantitative model for fault prediction was available (Robinson, 1993), as well as simulated data representing system faults and normal behavior. For each available datum, the model was used to predict the fault. This prediction was added as a 'feature' to the datum, as were various intermediate computations made by the model for the data point. The data points aug-

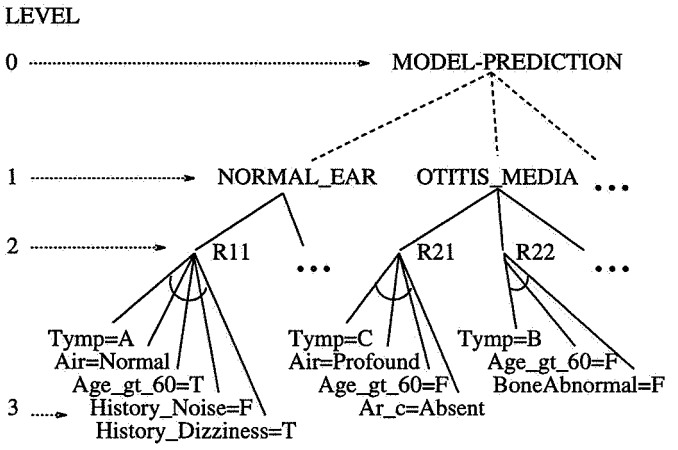

Figure 2: Levels in Audiology Theory

mented in this way were then given to C4.5, which constructed a classifier that predicted either a system's fault or normal operation. If the model were 'perfect' then we would expect that C4.5 would build a tree that only tested the model-based final prediction. Such a tree would indicate that if a new datum were encountered (represented by readings of various pressures and temperatures and other observables), then one should simply simulate the model on this datum and use the model-based final prediction. In the case of certain imperfections, a decision tree that tested various 'raw' features, as well as various model-based features might be constructed.

To our initial surprise, C4.5 consistently constructed trees that never or rarely referenced any model-based features. Rather than taking this as evidence of significant model imperfection or that the model added little or no information above and beyond that implicit in the raw features, a NASA analyst familiar with this application indicated that the simulated data used for training was unrepresentative or skewed – it represented a very small subspace of the RCS description space.

This work motivated an approach that weakly biases our adaptation of C4.5 to select model-based features. In particular, for purposes of this paper we assume a propositional domain theory used for classification that is acyclic and directed from the observable propositions to a final classification. A partial description of the perfect domain theory for the audiology domain used in our experiments is shown in Figure 2 as a tree. The domain theory

is a set of rules, each one consisting of a set of conditions together with the classification predicted by the rule. In Figure 2 the antecedents of a rule are listed at the leaves of the tree. Each condition is an attribute-value pair (e.g., Air=Profound). There may be several rules that predict a particular classification, as illustrated by the several possible rules leading to each classification (e.g., OTITIS-MEDIA) in Figure 2.

3.2 SIG: Implementing a Flexible Domain-Theory Preference Bias

To bias C4.5 towards model features closer in the hierarchy to the final model prediction, we order features according to their level number from the model prediction feature, through intermediate concept features, to rule features, and raw features. At each step during induction, our variation of C4.5 chooses a feature of smallest level number, unless a statistically-significant better feature (in terms of C4.5's information score) of larger level number is found. Hence, C4.5 will choose the model prediction feature unless sufficient evidence is present in the data to refute this choice.

Thus, we bias our inductive algorithm toward the model prediction feature and other features closer to it (of small level number). In a situation where we have a reasonably accurate model, and the available data is unrepresentative we expect our model-biased method to work better than a default strategy of choosing the feature of highest information value according to the available data (e.g., as in the standard C4.5). Nonetheless, if the data sufficiently contradicts the model, the model-bias can be abandoned and, should we choose, the model can be revised accordingly.

3.3 Using Domain-Theory Bias with Hypothesis Testing

The major difference between the original and the SIG variation of C4.5 is the manner in which a feature is selected for each node of a decision tree. C4.5 selects the feature with the highest information value according to the *information gain ratio* measure. Rather than selecting the feature with the highest information value outright, SIG requires that this value be statistically significantly higher than the information value of all features

preceding it in a feature preference ranking like that described in the previous section. Put in another way, we select the highest feature in a preference ranking that has an information score not significantly worse than any feature lower in the preference ranking.

The above procedure is implemented by the function **SelectFeature**($Node$), shown in Figure 3, where F_P is the feature preference ranking [1]; D is the set of training data associated with the current node; $info(F_j, D)$ is the value of C4.5's information measure for feature F_j when evaluated on the set of data D; and F_I is the list of the features sorted in descending order according to this measure. **SelectFeature**($Node$) initially chooses the feature with highest information value (i.e., first feature in F_I). However this feature is not accepted unless its information value is *significantly higher* than all features of higher preference, according to the F_P ranking. If so, the candidate feature is selected. Otherwise, the higher preference feature found becomes the new candidate. The procedure is repeated until a significant difference is found or the F_I list is exhausted.

There is also a minor difference between the classification procedure of our system and the standard C4.5 algorithm for the situations where there is insufficient data to select a test for a particular node of the tree. As a purely data driven system, the best C4.5 can do is to predict the most common class present in the current node. Instead, since we assume our model is better than no information, we use the prediction of our prior model.

The critical component of the function **SelectFeature** is the **Significantly-Better**(f_{cand}, f_{pref}, D) function. This function returns true if the information value of feature f_{cand} is estimated to be significantly higher than that of f_{pref}, according to a given level of statistical significance $SigLevel$. This is done by testing the null hypothesis that the difference between the information values of f_{cand} and f_{pref} is zero. If this null hypothesis can be rejected with $1 - SigLevel$ confidence **Significantly-Better** concludes that f_{cand} is significantly better than f_{pref}.

If the form of the probability distribution associated with C4.5's information measure is known and its parameters can be calculated, then traditional statistical theory can be used to test significance. This could be done for

[1] In the current implementation the ranking is a total ordering: features are sorted in ascending according to level number. The ranking of features within a level number is arbitrary.

Given: prior preference list $F_P = f_1, f_2, ..., f_F$

Function **SelectFeature**($Node$)

Set: D to set of observations in $Node$.
Create: list of features $F_I = f_{j_1}, f_{j_2}, ..., f_{j_{F_I}}$
sorted in descending order according to value of $info(f_{j_i}, D)$, eliminating any feature of null information value. In the case of nominal features this precludes the consideration of a feature used previously in the same path.
Set $f_{cand} = f_{j_1}$
While no significant difference has been found and there remain features to consider in F_I
Set f_{pref} to the first feature in F_I after f_{cand} that precedes f_{cand} in F_P. Eliminate all other features between f_{cand} and f_{pref} in F_I from consideration.
If Not(SignificantlyBetter(f_{cand}, f_{pref}, D))
Set $f_{cand} = f_{pref}$
EndIf
EndWhile
Return(f_{cand})

Figure 3: Function **SelectFeature**

the information gain measure, since Musick, Catlett, & Russell (1993) prove that this measure is normally distributed and provide explicit formulas for the parameters of this distribution. However, the form of the distribution for the default measure used in C4.5, information gain ratio, is not known. Fortunately, *Bootstrap Methods* (Efron & Gong, 1983) allow for estimates of significance levels of arbitrary statistics when the form and parameters of the underlying distribution are not known (Noreen, 1989). This is the method implemented in the function **Significantly-Better**.

In Efron's Bootstrap methods an unknown complete population P is estimated by repeated uniform subsampling with replacement from an available sample D of P. From D we obtain a set of bootstrap subsamples $P_B = \{D_1 ... D_{N_B}\}$, where N_B is a prespecified number of subsamples. Each D_i (with $1 \leq i \leq N_B$) is very likely to contain some duplicates and be missing some observations from D, with the result that the values of $info(f_j, D_i)$ for each feature F_j will likely be different on each bootstrap subsample D_i. Under some additional assumptions, we then proceed as if the bootstrap samples were obtained from the actual population P.

Significantly-Better uses two different bootstrap methods described by Noreen (1989): the Normal Approximation Method, and the Shift Method. Ortega and Fisher (1995) shows the computation of some quantities used in the above methods in greater detail.

We only decide that the feature f_{cand} is significantly better than f_{pref} if it is significantly better according to both the Normal Approximation Method and the Shift Method. **Significantly-Better** is computationally quite expensive. However, during the selection of most features this needs to be done very few times. If the feature with the highest initial information value is the feature of highest preference, **Significantly-Better** never needs to be computed. When other features are initially selected, only the features with higher preference are checked. As soon as one significant difference is computed, no other significance computation is necessary.

3.4 SIG: Concluding Remarks

Experiments have been performed that vary imperfection in a model, the representativeness of data, and the the veracity with which model-derived features are preferred. Ortega and Fisher (1995) and Ortega (1995a) have used hand- and machine- crafted propositional domain theories in these stud-

ies. Unfortunately, the RCS domain that motivated this research contained very skewed (unrepresentative) data to begin with, thus making it difficult to test SIG, C4.5, and other approaches on representative test sets.

In addition, Ortega (1995a) has developed an alternative method for combining information from multiple models (e.g., experts), multiple learning algorithms and data. The basic idea is to use induction over reencoded data. For each model, the data is reencoded and classified in terms of whether the model correctly or incorrectly classifies it. A 'referee' or 'judge' is then constructed from the data (using C4.5) that determines general conditions under which the model is to be believed or disbelieved. Collectively, the referees for the models are used to determine which model is to be believed for each test observation. Referees for 'models' constructed by different learning algorithms are also constructed using a cross-validation methodology. In general, Ortega's initial results suggest that this 'meta-learning' strategy is quite effective in combining information from multiple knowledge sources.

Ortega's work to date has used inductive techniques in conjunction with propositional theories, and to a lesser extent with Robinson's qualitative-quantitative model of the RCS system. Ortega is extending his earlier work in several directions. In particular, he is extending his techniques to interface induction with purely quantitative models. In particular, we have obtained a quantitative model of glucose/insulin dynamics in diabetics from the National Institute of Health. This model simulates glucose and insulin levels in a diabetic patient using equations that have been derived from 'averaged behaviors' over a large population of subjects. Ortega's goal is to take this general model of an 'average' patient, and adapt it to be a better predictor of glucose levels in a specific patient. Our goal is to use inductive techniques to discover conditions under which the model accurately reflects patient glucose levels, and in cases where it does not, to inductively-derive rules that generate better predictions of glucose levels than the model. Our long-term goal is the development of a 'knowledge-engineering' tool that diabetics can use to discover patterns of behavior that enable them to better manage their disease.

4 Basic Research on Cluster Analysis

Finally, work on clustering for operating mode discovery (Manganaris, Fisher, & Kulkarni, 1993) motivated some important extensions to the clustering strategy that we had used (i.e., Fisher's COBWEB which constructs an initial hierarchical clustering). One modification appends an iterative optimization technique onto the clustering system; this optimization strategy appears to be novel in the clustering literature – collections of observations are reclassified *en masse*, which appears to mitigate problems associated with local maxima. A second modification improves the noise tolerance of the clustering system. In particular, we adapt resampling-based pruning strategies used by supervised learning systems to the task of simplifying hierarchical clusterings, thus easying post-clustering analysis. Experiments confirm that hierarchical clusterings can be greatly simplified with no loss of significant information about patterns in the data.

4.1 Generating Hierarchical Clusterings

The clustering system used by Manganaris, Fisher, and Kulkarni was COBWEB. This section briefly summarizes the basic strategy (which we call hierarchical sorting) that they used, and then describes an iterative optimization procedure that we append to a clustering system. This optimization procedure yields a more robust approach that is more consistent in discovering patterns in data.

Clustering is a form of unsupervised learning that partitions observations into classes or clusters (collectively, called a clustering). Each observation is a vector of values along distinct observable variables. An objective function guides this search, ideally for a clustering that is optimal as measured by the objective function. A hierarchical clustering system creates a tree-structured clustering, where each set of sibling clusters partitions the observations covered by their common parent. This section briefly summarizes a very simple strategy, called *hierarchical sorting*, for creating hierarchical clusterings, and an iterative optimization strategy that we then apply to initial clusterings.

4.1.1 An Objective Function

We assume that an observation is a vector of nominal values, V_{ij} along distinct variables, A_i. A measure of *category utility* (Corter & Gluck, 1992), $CU(C_k) =$

$$P(C_k)\sum_i\sum_j[P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2],$$

has been used extensively by a system known as COBWEB (Fisher, 1987) and many related systems (e.g., Biswas, Weinberg, & Li, 1994).

In Fisher's (1987) COBWEB system, the quality of a partition of data is measured by $PU(\{C_1, C_2, \ldots C_N\}) = \sum_k CU(C_k)/N$ or the average category utility of clusters in the partition.

4.1.2 The Structure of Clusters

As in COBWEB, AUTOCLASS (Cheeseman, et. al., 1988), and other systems (Anderson & Matessa, 1991), we will assume that clusters, C_k, are described probabilistically: each variable value has an associated conditional probability, $P(A_i = V_{ij}|C_k)$, that reflects the proportion of observations in C_k that exhibit the value, V_{ij}, along variable A_i. In fact, each variable value is actually associated with the number of observations in the cluster having that value; probabilities are computed 'on demand' for purposes of evaluation. In addition, there is a single *root* cluster, identical in structure to other clusters, but covering all observations and containing frequency information necessary to compute $P(A_i = V_{ij})$'s as required by category utility. Clusters are arranged into a probablistic categorization tree (i.e., hierarchical clustering) in which each node is a cluster of observations summarized probabilistically. Observations are at leaves.

4.1.3 Hierarchical Sorting

Our strategy for initial clustering is *sorting*. Given an observation and a current partition, sorting evaluates the quality of new clusterings that result from placing the observation in each of the existing clusters, and the quality of the clustering that results from creating a new cluster that only covers the new observation; the option that yields the highest quality score (e.g., using

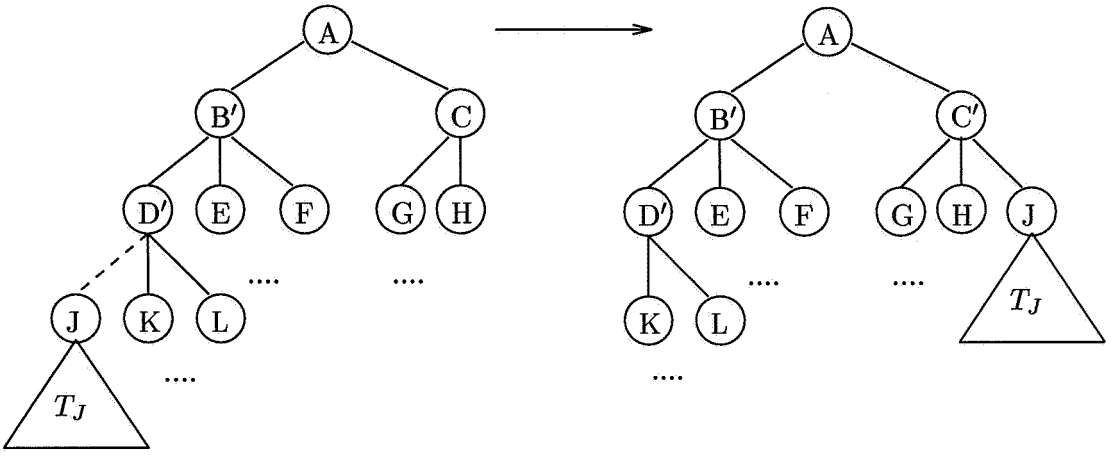

Figure 4: Hierarchical redistribution: the left subfigure indicates that cluster J has just been removed as a descendent of D and B, thus producing D' and B', and is about to be resorted relative to the children of the root (A). The rightmost figure shows J has been placed as a new child of C.

PU) is selected. The clustering grows incrementally as new observations are added.

This procedure is easily incorporated into a recursive loop that builds tree-structured clusterings: given an existing hierarchical clustering, an observation is sorted relative to the top-level partition (i.e., children of the root); if an existing child of the root is chosen to include the observation, then the observation is sorted relative to the children of this node, which now serves as the root in this recursive call. When a leaf is reached, the tree is extended downward. The maximum height of the tree can be bounded, thus limiting downward growth to fixed depth.

This sorting strategy is identical to that used by Anderson and Matessa (1991) and is a subset of the operations performed by COBWEB (Fisher, 1987).

4.1.4 Hierarchical Redistribution

Hierarchical sorting constructs a tree-structured clustering cheaply, but this greedy procedure typically constructs nonoptimal clusterings. Thus, after an initial clustering phase, a possibly offline process of iterative optimization

seeks to uncover better clusterings.

An iterative optimization strategy that appears novel in the clustering literature is *iterative hierarchical redistribution.* It can be contrasted with a very common strategy of redistributing single observations: after initial clustering, observations may be moved one at a time from one cluster to another, if to do so leads to an improved clustering according to the objective function. However, redistributing observations one at a time is very limited. In particular, the movement of an observation may be required for the eventual discovery of a better clustering, but the movement of any single observation may initially reduce clustering quality, thus preventing the discovery of the better clustering. In response, hierarchical redistribution considers the movement of observation *sets*, represented by existing clusters in a hierarchical clustering.

Given an existing hierarchical clustering, an outer recursive loop examines sibling clusters in the hierarchy in a depth-first fashion. For each set of siblings, an inner, iterative loop examines each, removes it from its current place in the hierarchy (along with its subtree), and resorts the cluster relative to the entire hierarchy. Removal requires that the various counts of ancestor clusters be decremented. Sorting the removed cluster is done based on the cluster's probabilistic description, and requires a minor generalization of the procedure for sorting individual observations: rather than incrementing certain variable value counts by 1 at a cluster to reflect the addition of a new observation, a 'host' cluster's variable value counts are incremented by the corresponding counts of the cluster (i.e., root of the subtree) being classified. A cluster may return to its original place in the hierarchy, or as Figure 4 illustrates, it (e.g., cluster J) may be sorted to an entirely different location.

The inner loop reclassifies each sibling of a set, and repeats until two consecutive iterations lead to the same set of siblings. The outer loop then turns its attention to the children of each of these remaining siblings. Eventually, the individual observations represented by leaves are resorted (relative to the entire hierarchy) until there are no changes from one iteration to the next. The outer loop may make several passes through the hierarchy until no changes occur from one pass to the next.

In sum, hierarchical redistribution takes large steps in the search for a better clustering. Similar to macro-operator learners (Iba, 1989) in problem-solving contexts, moving an observation set or cluster bridges distant points in the clustering space, so that a desirable change can be made that would

not otherwise have been viewed as desirable if redistribution was limited to movement of individual observations. The redistribution of increasingly smaller, more granular clusters (terminating with individual observations) serves to increasingly refine the clustering.

4.1.5 Results with Hierarchical Redistribution

Fisher (1995, 1996) evaluated hierarchical redistribution experimentally: a random ordering of observations is generated and hierarchically sorted. Hierarchical redistribution is then applied to the resultant hierarchical clustering. These experiments assume that the primary goal of clustering is to discover a single-level partition of the data that is of optimal quality. Thus, the *objective function score of the first-level partition* is taken as the most important dependent variable.

Table 2 shows results in 4 domains when the initial tree constructed by sorting is bounded to be no more than height 3 (i.e., the root has height 3, the leaves, which are single observations, are height 0, and there may be up to two levels of intermediate clusters). Row one for each domain shows the PU scores of initial clusterings and the time (in seconds) required to construct them.[2] Row two of each domain shows the PU scores after hierarchical redistribution, and the *additional* time required for this optimization process. In general, hierarchical redistribution consistently improves clustering quality in reasonable time. Fisher (1996) describes other experiments that (1) evaluate two alternative forms of iterative optimization, (2) evaluate optimization strategies using very 'poor' initial clusterings, and (3) evaluate clustering quality and the time required for optimization as one varies the height of the initial clustering. These experiments reveal that hierarchical redistribution is robust across all these dimensions and is superior, with caveats, to the alternative optimization strategies examined.

Hierarchical redistribution improves the results obtained with hierarchical sorting, but it may be appended to other greedy, hierarchical techniques as well, such as agglomerative clustering methods.

[2]Routines were implemented in SUN Common Lisp, compiled, and run on a SUN 3/60.

Table 2: Hierarchical redistribution with initial clusterings generated from sorting random ordered observations. Tree height is 3. Averages and standard deviations of PU scores and Time (seconds) over 20 trials.

		PU score	Time
Soybean/s	sort	1.53 (0.11)	18.3 (1.8)
47obs,36vars	hier.	1.62 (0.00)	93.8 (27.5)
Soybean/l	sort	0.89 (0.08)	142.4 (10.2)
307obs,36vars	hier.	1.07 (0.02)	436.3 (138.9)
House	sort	1.22 (0.30)	104.3 (8.7)
435obs,17vars	hier.	1.68 (0.00)	355.0 (71.1)
Mushroom	sort	1.10 (0.13)	406.6 (64.2)
1000obs,23vars	hier.	1.27 (0.00)	1288.2 (458)

4.2 Simplifying Hierarchical Clusterings

A hierarchical clustering can be grown to arbitrary depth. If there is structure in the data, then ideally the top layers of the clustering reflect this structure (and substructure as one descends the hierarchy). However, lower levels of the clustering may not reflect meaningful structure. Inspired by certain forms of retrospective pruning in decision-tree induction, we use resampling to identify 'frontiers' of a hierarchical clustering that are good candidates for pruning. Following initial hierarchy construction and iterative optimization, this simplification process is a final phase of search through the space of hierarchical clusterings that is intended to ease the burden of a data analyst.

4.2.1 Identifying Variable Frontiers

Several authors (Fisher, 1987; Cheeseman, et. al., 1988; Anderson & Matessa, 1991) motivate clustering as a means of improving performance on a task akin to pattern completion, where the error rate over completed patterns can be used to 'externally' judge the utility of a clustering. Given a probabilistic categorization tree, a new observation with an unknown value for a variable can be classified down the hierarchy using a small variation on the hierarchical sorting procedure described earlier. Classification is terminated at an existing node (cluster) along the classification path, and the variable value of

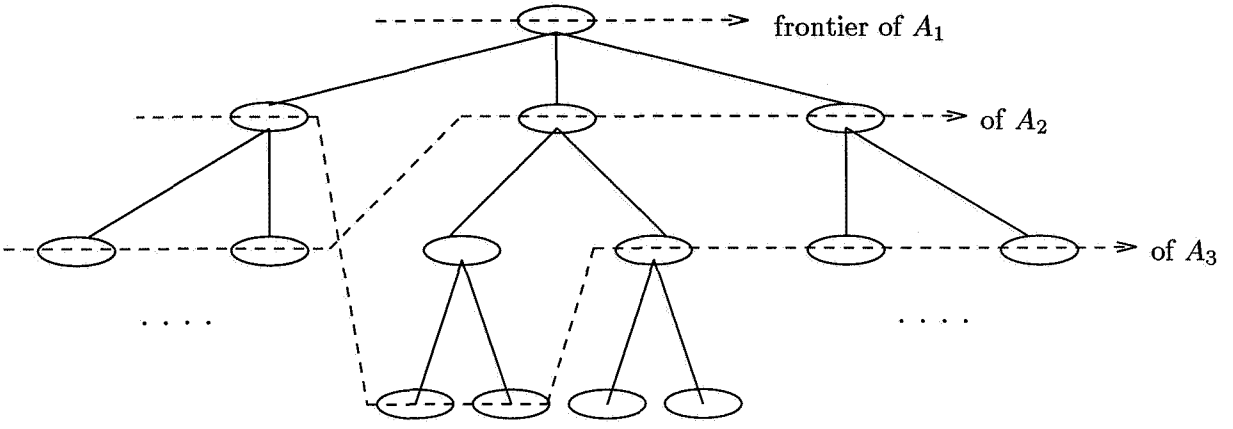

Figure 5: Frontiers for three variables in a hypothetical clustering.

highest probability at that cluster is predicted as the unknown variable value of the new observation. Naively, classification might always terminate at a leaf (i.e., an observation), and the leaf's value along the specified variable would be predicted as the variable value of the new observation. However, a variable might be better predicted at some internal node in the classification path. We adapt retrospective pruning strategies in decision tree induction, such as *reduced error pruning* (Quinlan, 1987), to the task of identifying these internal nodes.

Given a hierarchical clustering and a *validation* set of observations, the validation set is used to identify an appropriate *frontier* of clusters for prediction of each variable. Figure 5 illustrates that the preferred frontiers of any two variables may differ, and clusters within a frontier may be at different depths. For *each* variable, A_i, the objects from the validation set are each classified through the hierarchical clustering with the value of variable A_i 'masked' for purposes of classification. At each cluster encountered during classification the observation's value for A_i is compared to the most probable value for A_i at the cluster; if they are the same, then the observation's value would have been correctly predicted at the cluster. A count of all such correct predictions for each variable at each cluster is maintained. Following classification for all variables over all observations of the validation set, a preferred frontier for each variable is identified that maximizes the number of correct counts for the variable.

The identification of variable-specific frontiers facilitates a number of pruning strategies. For example, a node that lies below the frontier of every variable offers no advantage in terms of pattern-completion error rate; such a node probably reflects no meaningful structure and it (and its descendents) may be pruned. However, if an analyst is focusing attention on a subset of the variables, then frontiers might be more flexibly exploited for pruning.

The novelty of the validation strategy described here stems from an observation that any single partition of observations may overfit the data relative to some variables and underfit relative to others. Undoubtedly, the identification of variable-specific frontiers can also be implemented by adapting Bayesian or hypothesis-testing techniques, which are currently used to terminate hierarchical decomposition by identifying a single, variable-independent frontier (e.g., AUTOCLASS).

4.2.2 Experiments with Validation

To test the validation procedure's promise for simplifying hierarchical clusterings, each of the data sets used in the experiments of Section 3.6.5 was randomly divided into three subsets: 40% for training, 40% for validation, and 20% for test. A hierarchical clustering is first constructed by sorting the training set. This hierarchy is then optimized using hierarchical redistribution. The final hierarchy decomposes the training set to singleton clusters, each containing a single training observation. The validation set is then used to identify variable frontiers within the entire hierarchy.

During testing of a validated clustering, each variable of each test observation is masked in turn. When classification reaches a cluster on the frontier of the masked variable, the most probable value is predicted as the value of the observation; the proportion of correct predictions for each variable over the test set is recorded. For comparative purposes, we also use the test set to evaluate predictions stemming from the unvalidated tree, where all variable predictions are made at the leaves (singleton clusters) of this tree.

Table 3 shows results from 20 experimental trials using unvalidated and validated clusterings. The first row of each domain lists the average number of leaves for the unvalidated and validated trees. The unvalidated clusterings decompose the training data to single-observation leaves – the number of leaves equals the number of training observations. In the validated clustering, we assume that clusters are pruned if they lie below the frontiers of

Table 3: Characteristics of optimized clusterings before and after validation. Average and standard deviations over 20 trials.

		Unvalidated	Validated
Soy/s	Leaves	18.00 (0.00)	13.10 (1.59)
	Accuracy	0.85 (0.01)	0.85 (0.01)
	|Frontier|	18.00 (0.00)	2.75 (1.17)
Soy/l	Leaves	122.00 (0.00)	79.10 (5.80)
	Accuracy	0.83 (0.02)	0.83 (0.02)
	|Frontier|	122.00 (0.00)	17.01 (4.75)
House	Leaves	174.00 (0.00)	49.10 (7.18)
	Accuracy	0.76 (0.02)	0.81 (0.01)
	|Frontier|	174.00 (0.00)	9.90 (5.16)
Mush	Leaves	400.00 (0.00)	96.30 (11.79)
	Accuracy	0.80 (0.01)	0.82 (0.01)
	|Frontier|	400.00 (0.00)	11.07 (4.28)

all variables. Thus, a leaf in a validated clustering is a cluster (in the original clustering) that is on the frontier of *at least one* variable, and none of its descendent clusters (in the original clustering) are on the frontier of any variable.

Prediction accuracies in the second row of each domain entry are the mean proportion of correct predictions over *all* variables over 20 trials. Predictions were generated at leaves (singleton clusters) in the unvalidated hierarchical clusterings and at appropriate variable frontiers in the validated clusterings. In all cases, validation/pruning substantially reduces clustering size and it does not diminish accuracy.

We have suggested that more flexible pruning or 'attention' strategies might be possible when an analyst is focusing on one or a few variables. We will not specify such strategies, but the statistic given in row 3 of each domain entry suggests that clusterings can be rendered in considerably simpler forms when an analyst's attention is selective. Row 3, labeled |Frontier|, is the *average number of frontier clusters per variable.* This is an average over all variables and all experimental trials. Intuitively, a frontier cluster of a variable is a 'leaf' as far as prediction of that variable is concerned. The

|Frontier| entry for unvalidated clusterings is simply given by the number of leaves, since this is where all variable predictions are made in the unvalidated case. Our results suggest that when attention is selective, a partial clustering that captures the structure involving selected variables can be presented to an analyst in very simplified form.[3]

4.3 Concluding Remarks

The work of Manganaris, Fisher, and Kulkarni (1993), in part, motivated improvements to a clustering system that was at the core of their approach. We can abstract the work on clustering described here though away from the NASA-application work that motivated it. In general, hierarchical redistribution makes an aggressive search for clusterings that are optimal relative to objective function, and retrospective pruning removes those parts of a clustering that do not represent meaningful structure in data. For purposes of evaluation, Fisher (1995, 1996) coupled these with a particular objective function, initial clustering strategy, and data representation. These two strategies, however, may be coupled with other initial clustering strategies, objective functions, and (e.g., numeric) data representations as well.

5 Other Applications and Research

NASA Ames NAG 2-834 has also supported work, in part, that extends inductive learning methods to industrial and management applications. In particular, Srinivasan and Fisher (1995) have used machine induction to estimate software development effort from characteristics of a software specification. This application requires that data (e.g., software projects) be 'classified' along a continuous dimension (person-month required to develop the software). One of the basic techniques is a model for the type of learning system that Ortega's plans for the diabetic management task.

Saraf and Fisher (in press) have used induction methods for road-traffic signal control. The data (traffic 'demands') is temporal, and may serve as an additional applied domain in which to apply CALCHAS.

[3]Fisher (1996) also looks at the relative amount of simplification that can be performed with optimized (using hierarchical redistribution) and unoptimized clusterings.

6 Concluding Remarks

In sum, NAG 2-834 has supported three primary lines of research on learning from temporal data, on combining inductive and model-based reasoning, and clustering research, as well as partially supporting the P.I. on some auxiliary application/research in traffic control and software characterization. We have noted some future directions in the case of the primary research areas, but our experience in industrial applications, primarily a printing application, suggests a promising area of research as well. In particular, Evans and Fisher (1994) adapted decision tree induction to diagnose certain process delays at a large U. S. printing plant. The application highlighted an important issue that undoubtedly arises in NASA applications as well. In particular, rules that are learned inductively are not simply intended to describe the environment, but the ultimate goal is to use these rules prescriptively, to direct the environment towards desirable behaviors. However, in real applications not all dimensions that are used to describe data can be easily manipulated/controlled; rules that include these features may be predictive of a certain behavior, but it may be impossible or difficult to 'apply' this rule to modify the environment. For example, in the printing application, plant humidity turned out to be predictive of printing quality, but it is not cost-effective to humidify the plant. Other researchers have looked at the cost of making measurements along certain dimensions, but we would like to take these ideas further, by taking into account the cost of manipulating dimensions. Rules that are predictive of a desirable outcome, but also include features that can be manipulated are the most desirable kinds of rules in control contexts.

7 NAG 2-834 supported Publications

Fisher, D. (1996). Iterative optimization and validation of hierarchical clusterings. *Journal of Artificial Intelligence Research, 4*, 147–178.

Fisher, D. (1995). Optimization and validation of hierarchical clusterings. *First International Conference on Knowledge Discovery & Data Mining*, (pp. 118–123), Montreal, Ontario, Canada.

Manganaris, S. (1994). *Supervised learning of time series.* Technical Report 94-02, Department of Computer Science, Vanderbilt University.

Manganaris, S. (1995a). Bayesian induction of features in temporal domains. *IJCAI-95 Workshop on Data Engineering for Inductive Learning*, Montreal, Ontario, Canada.

Manganaris, S. (1995b). Learning to classify sensor data. *IJCAI-95 Workshop on Machine Learning in Engineering*, Montreal, Ontario, Canada.

Manganaris, S. (in press). Classifying Sensor Data with CALCHUS. *International Journal of Intelligent Real-Time Automation.*

Manganaris, S., & Fisher, D. (1994). Learning time series for intelligent monitoring. *i-SAIRAS*, (pp. 71–74), Pasadena, CA.

Manganaris, S., Fisher, D., & Kulkarni, D. (1993). Discovering operating modes in telemetry data from the shuttle reaction control system. *Proceedings of the SOAR Conference*, (pp. 234–244), Houston, TX.

Ortega, J. (1995a). *Making the most of what you've got: Using models and data to improve learning rate and prediction accuracy.* Doctoral Dissertation. Department of Computer Science, Vanderbilt University, Nashville, TN.

Ortega, J. (1995b). On the informativeness of the DNA Promoter Sequences Domain Theory. *Journal of Artificial Intelligence Research, 2*, 361–367.

Ortega, J., & Fisher, D. (1995). Flexibly exploiting prior knowledge in empirical learning. *International Joint Conference on Artificial Intelligence*, (pp. 1041–1047). Montreal, Ontario, Canada.

Saraf, R., & Fisher, D. (in press). On-line signal plan generation for centralized control using neural networks. *IVHS Journal.*

Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development time. *IEEE Transactions on Software Engineering, 21*, 126–138.

8 Other References

Anderson, J. R., & Matessa, M. (1991). An iterative Bayesian algorithm for categorization. In Fisher, D., Pazzani, M., & Langley, P. (Eds.), *Concept Formation: Knowledge and Experience in Unsupervised Learning.* San Mateo, CA: Morgan Kaufmann.

Cheeseman, P. (1990). On finding the most probably model. In J. Shrager and P. Langley (Eds.), *Computational Models of Discovery and Theory Formation*, Morgan Kaufmann.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. In *Proceedings of the Fifth International Machine Learning Conference*, pp. 54-64. Ann Arbor, MI: Morgan Kaufmann.

Corter, J., & Gluck, M. (1992). Explaining basic categories: feature predictability and information. *Psychological Bulletin, 111*, 291-303.

Drastal, G., Czako, G., & Raatz, S. (1989). Induction in an abstraction space: a form of constructive induction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, (pp. 708–712), Detroit, MI.

Evans, R., & Fisher, D. (1994). Process delay analysis using decision tree induction. *IEEE Expert, 9*, 60–66.

Efron, B., & Gong, G. (1983). A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician, 37*, 36–48.

Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Iba, G. (1989). A heuristic approach to the discovery of macro operators. *Machine Learning, 3*, 285–317.

Jaynes, E. (1993). *Probability Theory: The Logic of Science.* Advance fragments of a book in preparation made available by author.

Musick, R., Catlett, J., & Russell, S. (1993). Decision theoretic subsampling for induction on large databases. In *Proceedings of the Tenth International Conference on Machine Learning*, (pp. 212–219), Amherst, MA.

Noreen, E. (1989). *Computer Intensive Methods for Testing Hypotheses: An Introduction.* John Wiley & Sons, New York, NY.

Pednault, E. (1989). Some experiments in applying inductive inference principles to surface reconstruction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, (pp. 1603–1609), Detroit, MI.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies, 27*, 221–234.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Rao, B., & Lu, S. (1992). Learning engineering models with the minimum description length principle. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, (pp. 717-722), San Jose, CA.

Rendell, L., & Seshu, R. (1990). Learning hard concepts through constructive induction: framework and rationale. *Computational Intelligence, 6*, 247–270.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *The Annals of Statistics, 11*, 416–431.

Robinson, P. (1993). *Automated fault diagnosis algorithms for the reaction control system of the space shuttle.* Technical Report FIA-93-05, NASA Ames Research Center.

Wallace, C., & Freeman, P. (1987). Estimation and inference by compact coding. *The Journal of the Royal Statistical Society, 49*, 252–265.

Acknowledgements

I thank Phil Laird, Ron Saul, (of NASA Ames Research Center) and Robert Shelton (Johnson Space Center) for providing the EGIL data used in the CALCHAS experiments. I also thank Deepak Kulkarni and Peter Robinson (of NASA Ames Research Center) for early discussion on initial strategies that led to the SIG system.